WEST Search History

Hide Items | Restore | Clear | Cancel

DATE: Monday, May 23, 2005

Hide?	Set Name	Query	Hit Count
	DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=ADJ		
☐	L6	L4 and L1	50
	DB=EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=ADJ		
☐	L5	L4	1
	DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=ADJ		
☐	L4	L3 and optimiz$	87
☐	L3	L2 and (hot spot or hotspot)	128
☐	L2	(Virtual machine or JIT) and profil$	2043
☐	L1	717/124-167.ccls.	4972

END OF SEARCH HISTORY

Terms used **profile** **virtual machine** **hotspot** **cache miss** Found **18** of **154,226**

Sort results by: relevance
Display results: expanded form
Save results to a Binder
Search Tips
☐ Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 18 of 18

Relevance scale

1 Profiling Java applications using code hotswapping and dynamic call graph revelation
Mikhail Dmitriev
January 2004 **ACM SIGSOFT Software Engineering Notes , Proceedings of the fourth international workshop on Software and performance**, Volume 29 Issue 1
Full text available: pdf(1.32 MB) Additional Information: full citation, abstract, references

Instrumentation-based profiling has many advantages and one serious disadvantage: usually high performance overhead. This overhead can be substantially reduced if only a small part of the target application (for example, one that has previously been identified as a performance bottleneck) is instrumented, while the rest of the application code continues to run at full speed. The value of such a profiling technology would increase further if the code could be instrumented and de-instrumented as m ...

2 Dynamic hot data stream prefetching for general-purpose programs
Trishul M. Chilimbi, Martin Hirzel
May 2002 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2002 Conference on Programming language design and implementation**, Volume 37 Issue 5
Full text available: pdf(210.85 KB) Additional Information: full citation, abstract, references, citings, index terms

Prefetching data ahead of use has the potential to tolerate the growing processor-memory performance gap by overlapping long latency memory accesses with useful computation. While sophisticated prefetching techniques have been automated for limited domains, such as scientific codes that access dense arrays in loop nests, a similar level of success has eluded general-purpose programs, especially pointer-chasing codes written in languages such as C and C++. We address this problem by describing ...

Keywords: data reference profiling, dynamic optimization, dynamic profiling, memory performance optimization, prefetching, temporal profiling

3 Concurrent garbage collection using hardware-assisted profiling
Timothy H. Heil, James E. Smith
October 2000 **ACM SIGPLAN Notices , Proceedings of the 2nd international symposium on Memory management**, Volume 36 Issue 1
Full text available: pdf(1.74 MB) Additional Information: full citation, abstract, citings, index terms

In the presence of on-chip multithreading, a Virtual Machine (VM) implementation can readily take advantage of *service threads* for enhancing performance by performing tasks such as profile collection and analysis, dynamic optimization, and garbage collection concurrently with program execution. In this context, a hardware-assisted profiling mechanism is proposed. The *Relational Profiling Architecture* (RPA) is designed from the top down. RPA is based on a relational model similar ...

4 Characterizing the memory behavior of Java workloads: a structured view and opportunities for optimizations
Yefim Shuf, Mauricio J. Serrano, Manish Gupta, Jaswinder Pal Singh
June 2001 **ACM SIGMETRICS Performance Evaluation Review , Proceedings of the 2001 ACM SIGMETRICS international conference on Measurement and modeling of computer systems**, Volume 29 Issue 1
Full text available: pdf(1.55 MB) Additional Information: full citation, abstract, references, citings

This paper studies the memory behavior of important Java workloads used in benchmarking Java Virtual Machines (JVMs), based on instrumentation of both application and library code in a state-of-the-art JVM, and provides structured information about these workloads to help guide systems' design. We begin by characterizing the inherent memory behavior of the benchmarks, such as information on the breakup of heap accesses among different categories and on the hotness of references to fields and met ...

5 Creating and preserving locality of java applications at allocation and garbage collection times
Yefim Shuf, Manish Gupta, Hubertus Franke, Andrew Appel, Jaswinder Pal Singh
November 2002 **ACM SIGPLAN Notices , Proceedings of the 17th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 37 Issue 11
Full text available: pdf(180.20 KB) Additional Information: full citation, abstract, references, citings, index terms

The growing gap between processor and memory speeds is motivating the need for optimization strategies that improve data locality. A major challenge is to devise techniques suitable for pointer-intensive applications. This paper presents two techniques aimed at improving the memory behavior of pointer-intensive applications with dynamic memory allocation, such as those written in Java. First, we present an allocation time object placement technique based on the recently introduced notion of *p* ...

***Keywords:** JVM, Java, garbage collection, heap traversal, locality, locality based graph traversal, memory allocation, memory management, object co-allocation, object placement, prolific types, run-time systems*

6 The Jrpm system for dynamically parallelizing Java programs
Michael K. Chen, Kunle Olukotun
May 2003 **ACM SIGARCH Computer Architecture News , Proceedings of the 30th annual international symposium on Computer architecture**, Volume 31 Issue 2
Full text available: pdf(320.42 KB) Additional Information: full citation, abstract, references, citings

We describe the Java runtime parallelizing machine (Jrpm), a complete system for parallelizing sequential programs automatically. Jrpm is based on a chip multiprocessor (CMP) with thread-level speculation (TLS) support. CMPs have low sharing and communication costs relative to traditional multiprocessors, and thread-level speculation (TLS) simplifies program parallelization by allowing us to parallelize optimistically without violating correct sequential program behavior. Using a Java virtual ma ...

7 A framework for reducing the cost of instrumented code
Matthew Arnold, Barbara G. Ryder
May 2001 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2001 conference on Programming language design and implementation**, Volume 36 Issue 5
Full text available: pdf(1.78 MB) Additional Information: full citation, abstract, references, citings, index terms

Instrumenting code to collect profiling information can cause substantial execution overhead. This overhead makes instrumentation difficult to perform at runtime, often preventing many known *offline* feedback-directed optimizations from being used in online systems. This paper presents a general framework for performing *instrumentation sampling* to reduce the overhead of previously expensive instrumentation. The framework is simple and effective, using code-duplication and *coun ...*

8 Continuous program optimization: A case study
Thomas Kistler, Michael Franz
July 2003 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 25 Issue 4
Full text available: pdf(877.67 KB) Additional Information: full citation, abstract, references, index terms, review

Much of the software in everyday operation is not making optimal use of the hardware on which it actually runs. Among the reasons for this discrepancy are hardware/software mismatches, modularization overheads introduced by software engineering considerations, and the inability of systems to adapt to users' behaviors.A solution to these problems is to delay code generation until load time. This is the earliest point at which a piece of software can be fine-tuned to the actual capabilities of the ...

Keywords: Dynamic code generation, continuous program optimization, dynamic reoptimization

9 Practicing JUDO: Java under dynamic optimizations
Michał Cierniak, Guei-Yuan Lueh, James M. Stichnoth
May 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2000 conference on Programming language design and implementation**, Volume 35 Issue 5
Full text available: pdf(190.06 KB) Additional Information: full citation, abstract, references, citings, index terms

A high-performance implementation of a Java Virtual Machine (JVM) consists of efficient implementation of Just-In-Time (JIT) compilation, exception handling, synchronization mechanism, and garbage collection (GC). These components are tightly coupled to achieve high performance. In this paper, we present some static anddynamic techniques implemented in the JIT compilation and exception handling of the Microprocessor Research Lab Virtual Machine (MRL VM), ...

10 Online feedback-directed optimization of Java
Matthew Arnold, Michael Hind, Barbara G. Ryder
November 2002 **ACM SIGPLAN Notices , Proceedings of the 17th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 37 Issue 11
Full text available: pdf(463.00 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper describes the implementation of an online feedback-directed optimization system. The system is fully automatic; it requires no prior (offline) profiling run. It uses a previously developed low-overhead instrumentation sampling framework to collect control flow graph edge profiles. This profile information is used to drive several traditional optimizations, as well as a novel algorithm for performing feedback-directed control flow graph node splitting. We empirically evaluate this syst ...

Keywords: adaptive optimization, dynamic optimization, online algorithms, virtual machines

11 Improving Java performance using hardware translation
Ramesh Radhakrishnan, Ravi Bhargava, Lizy K. John
June 2001 **Proceedings of the 15th international conference on Supercomputing**
Full text available: pdf(254.91 KB) Additional Information: full citation, abstract, references, citings, index

terms

State of the art Java Virtual Machines with Just-In-Time (JIT) compilers make use of advanced compiler techniques, run-time profiling and adaptive compilation to improve performance. However, these techniques for alleviating performance bottlenecks are more effective in long running workloads, such as server applications. Short running Java programs, or client workloads, spend a large fraction of their execution time in compilation instead of useful execution when run using JIT compilers. In ...

12 The Performance of Runtime Data Cache Prefetching in a Dynamic Optimization System
Jiwei Lu, Howard Chen, Rao Fu, Wei-Chung Hsu, Bobbie Othmer, Pen-Chung Yew, Dong-Yuan Chen
December 2003 **Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture**
Full text available: pdf(253.79 KB) Additional Information: full citation, abstract, citings, index terms

Traditional software controlled data cache prefetching isoften ineffective due to the lack of runtime cache miss andmiss address information. To overcome this limitation, weimplement runtime data cache prefetching in the dynamicoptimization system ADORE (ADaptive Object code RE-optimization).Its performance has been compared withstatic software prefetching on the SPEC2000 benchmarksuite. Runtime cache prefetching shows better performance.On an Itanium 2 based Linux workstation, it can increasepe ...

13 Dynamic metrics for java
Bruno Dufour, Karel Driesen, Laurie Hendren, Clark Verbrugge
October 2003 **ACM SIGPLAN Notices , Proceedings of the 18th annual ACM SIGPLAN conference on Object-oriented programing, systems, languages, and applications**, Volume 38 Issue 11
Full text available: pdf(222.67 KB) Additional Information: full citation, abstract, references, citings, index terms

In order to perform meaningful experiments in optimizing compilation and run-time system design, researchers usually rely on a suite of benchmark programs of interest to the optimization technique under consideration. Programs are described as *numeric*, *memory-intensive*, *concurrent*, or *object-oriented*, based on a qualitative appraisal, in some cases with little justification. We believe it is beneficial to quantify the behaviour of programs with a concise and precisely ...

Keywords: Java, dynamic metrics, execution traces, optimization, profiling, program analysis, software metrics

14 A brief history of just-in-time
John Aycock
June 2003 **ACM Computing Surveys (CSUR)**, Volume 35 Issue 2
Full text available: pdf(171.09 KB) Additional Information: full citation, abstract, references, citings, index terms

Software systems have been using "just-in-time" compilation (JIT) techniques since the 1960s. Broadly, JIT compilation includes any translation performed dynamically, after a program has started execution. We examine the motivation behind JIT compilation and constraints imposed on JIT compilation systems, and present a classification scheme for such systems. This classification emerges as we survey forty years of JIT work, from 1960--2000.

Keywords: Just-in-time compilation, dynamic compilation

15
SAFKASI: a security mechanism for language-based systems

Dan S. Wallach, Andrew W. Appel, Edward W. Felten
October 2000 **ACM Transactions on Software Engineering and Methodology (TOSEM)**, Volume 9 Issue 4
Full text available: pdf(234.89 KB) Additional Information: full citation, abstract, references, citings, index terms

In order to run untrusted code in the same process as trusted code, there must be a mechanism to allow dangerous calls to determine if their caller is authorized to exercise the privilege of using the dangerous routine. Java systems have adopted a technique called stack inspection to address this concern. But its original definition, in terms of searching stack frames, had an unclear relationship to the actual achievement of security, overconstrained the implementation of a Java system, lim ...

Keywords: Internet, Java, WWW, access control, applets, security-passing style, stack inspection

16 Research papers III: Comparative performance analysis of mobile runtimes on Intel XScale® technology
Jason Domer, Murthi Nanja, Suresh Srinivas, Bhaktha Keshavachar
June 2004 **Proceedings of the 2004 workshop on Interpreters, virtual machines and emulators**
Full text available: pdf(226.94 KB) Additional Information: full citation, abstract, references, index terms

Mobile Runtime Environments such as Java*2 Micro Edition (J2ME*) and Microsoft WinCE.NET* Compact Framework* are becoming standard managed application execution environments on memory constrained devices. A variety of implementations exists, and so too are a variety of systems they could run on, and finally a variety of workloads. It becomes important to understand how they compare.In this paper we describe comparative performance analysis of mobile runtimes on products with Intel XScale® mi ...

17 Hardware Support for Control Transfers in Code Caches
Ho-Seop Kim, James E. Smith
December 2003 **Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture**
Full text available: pdf(315.74 KB) Additional Information: full citation, abstract, index terms

Many dynamic optimization and/or binary translationsystems hold optimized/translated superblocks in a codecache. Conventional code caching systems suffer fromoverheads when control is transferred from one cachedsuperblock to another, especially via register-indirectjumps. The basic problem is that instruction addresses inthe code cache are different from those in the original programbinary. Therefore, performance for register-indirectjumps depends on the ability to translate efficiently fromsour ...

18 Reducing virtual call overheads in a Java VM just-in-time compiler
Junpyo Lee, Byung-Sun Yang, Suhyun Kim, Kemal Ebcioğlu, Erik Altman, Seungil Lee, Yoo C. Chung, Heungbok Lee, Je Hyung Lee, Soo-Mook Moon
March 2000 **ACM SIGARCH Computer Architecture News**, Volume 28 Issue 1
Full text available: pdf(994.66 KB) Additional Information: full citation, abstract, index terms

Java, an object-oriented language, uses *virtual methods* to support the extension and reuse of classes. Unfortunately, virtual method calls affect performance and thus require an efficient implementation, especially when just-in-time (JIT) compilation is done. *Inline caches* and *type feedback* are solutions used by compilers for dynamically-typed object-oriented languages such as SELF [1, 2, 3], where virtual call overheads are much more critical to performance than in Java. Wi ...

Keywords: Java JIT compilation, adaptive compilation, inline cache, type feedback, virtual method call

Results 1 - 18 of 18

Web Results **1** - **10** of about **79** for profile "virtual machine" hotspot "cache miss". (**0.58** seconds)

Paper: A Study of Cache Performance in Java Virtual Machines ...
... the Latte **Virtual Machine**, the cache simulator and configurations, ...
Execution engine **cache miss** rates under s1 and s100 data sets (Cache ...
computing.breinestorm.net/ cache+data+performance+execution+java/ - 53k - Cached - Similar pages

[PDF] A Study of Cache Performance in Java Virtual Machines
File Format: PDF/Adobe Acrobat - View as HTML
... source **virtual machine** released in October 1999, and was ... to configuration
3, average **cache miss** reductions are 47%. for reads in the execution phase ...
www.ece.utexas.edu/~jrubio/pubs/WWC02_anand.pdf - Similar pages

[PS] Execution Characteristics of Just-In-Time Compilers Technical ...
File Format: Adobe PostScript - View as Text
... gcc is shown to have much higher instruction **cache miss** rates as compared to
other C benchmarks. ... [12] "**HotSpot**: A new breed of **virtual machine**", ...
www.ece.utexas.edu/~jrubio/pubs/tr990717.ps - Similar pages

[PDF] Diagnosing Performance Overheads in the Xen **Virtual Machine** ...
File Format: PDF/Adobe Acrobat - View as HTML
... An application's performance in a **virtual machine** environment can differ markedly
... respectively), higher L2 **cache miss** rates (1.97 times higher ...
www.hpl.hp.com/techreports/2005/HPL-2005-80.pdf - Similar pages

[PDF] Diagnosing Performance Overheads in the Xen **Virtual Machine** ...
File Format: PDF/Adobe Acrobat - View as HTML
... application's performance in a **virtual machine** environ- ... respectively),
higher L2 **cache miss** rates (1.97 times higher. and 1.52 times higher), ...
www.hpl.hp.com/research/ dca/system/papers/xenoprof-vee05.pdf - Similar pages

7 - CHAPTER -
... memory management, and Java **HotSpot virtual machine** compilation. ... An example
of this sort of delay is a load instruction that has a data **cache miss**. ...
docs.sun.com/source/817-0922/AdvancedTopics.html - 135k - May 21, 2005 - Cached - Similar pages

7 - CHAPTER -
... on methods that are compiled by the Java **HotSpot** trademark **virtual machine**,
... of this sort of delay is a load instruction that has a data **cache miss**. ...
docs.sun.com/source/816-2458/AdvancedTopics.html - 105k - Cached - Similar pages

[PDF] Dynamic ProfileGuided OptimizationinaVEEonIA-64
File Format: PDF/Adobe Acrobat - View as HTML
... Weblogic JRockit, a Java **virtual machine**. To collect the **profile** information
the hardware ... Java 1.4.1 **HotSpot Virtual Machine**, White Paper. Sun ...
web.it.kth.se/~schulte/teaching/ theses/IMIT-LECS-2004-69.pdf - Similar pages

[PDF] Characterizing the Memory Behavior of Java Workloads: A Structured ...
File Format: PDF/Adobe Acrobat
... The L2 **cache miss** rates of Java workloads are very close to the ...
level characterization of Java **virtual machine** workload. In 2nd Annual ...
dx.doi.org/10.1145/378420.378783 - Similar pages

[PDF] A CO-DESIGNED **VIRTUAL MACHINE** FOR INSTRUCTION-LEVEL DISTRIBUTED ...

File Format: PDF/Adobe Acrobat - View as HTML
... Figure 1-3 Overview of the co-designed **virtual machine** in the thesis . ...
On an I-**cache miss**, translated ILDP instructions are fetched from the ...
www.ece.wisc.edu/~jes/papers/kim.thesis.pdf - Similar pages

Gooooooooogle ▶

Result Page: **1** 2 3 4 5 6 7 8 **Next**

Free! Get the Google Toolbar. Download Now - About Toolbar

Google Search 377 blocked Check AutoLink AutoFill

profile "virtual machine" hotspot "ca Search

Search within results | Language Tools | Search Tips | Dissatisfied? Help us improve

Google Home - Advertising Programs - Business Solutions - About Google

CiteSeer Find: profile AND virtual machine AND ho Documents Citations

Searching for **profile and virtual machine and hotspot.**
Restrict to: Header Title Order by: Expected citations Hubs Usage Date Try: Google (CiteSeer) Google (Web) Yahoo! MSN CSB DBLP
14 documents found. **Order: number of citations.**

Welcome to the Opportunities of Binary Translation - Altman, al. (2000) (Correct) (5 citations)
Static translators can also use execution **profiles** obtained during a program's previous run. All
can be ported to any platform for which a Java **virtual machine** (VM) is available. One approach to
with that of commercial Java VMs, such as Sun's **Hotspot** and JDK 1.2. Latte is a research prototype and
www.eecs.harvard.edu/cs253/papers/r3040.pdf

Online Feedback-Directed Optimization of Java - Matthew Arnold Michael (2002) (Correct) (2 citations)
framework to collect control flow graph edge **profiles**. This **profile** information is used to drive
www.research.rutgers.edu/~marnold/papers/oopsla02.ps

Application of the HotSwap Technology to Advanced Profiling - Dmitriev (2002) (Correct) (2 citations)
bytecode instrumentation would allow developers to **profile** dynamically selected parts of the target
implemented in the Sun's Java **HotSpot** tm **Virtual Machine**. This functionality was initially considered
has been recently implemented in the Sun's Java **HotSpot** tm **Virtual Machine**. This functionality was
www.dcs.gla.ac.uk/~misha/papers/hsprof.ps.gz

The Case for Multiple Compilers - Detlefs, Agesen (1999) (Correct) (2 citations)
different characteristics: some have very "steep" **profiles**, with almost all execution time spent in a
Presented at OOPSLA '99 **Virtual Machine** Workshop The Case for Multiple Compilers
More aggressive Sun, Sun Microsystems, Java, **HotSpot**, Solaris, and Ultra are trademarks or
wwwwswest.sun.com/research/people/detlefs/papers/99-oopsla-3tier.ps

Design, Implementation and Evaluation of Adaptive Recompilation .. - Fink, Qian (2003) (Correct) (1 citation)
including a study of fully automatic, online, **profile**-driven deferred compilation. 1 Introduction
fqian@sable.mcgill.ca Abstract Modern **virtual machines** often maintain multiple compiled versions of
VMs, with the notable exception of Sun's **HotSpot** Server Compiler [12]due to the perceived
www.research.ibm.com/people/s/sfink/papers/cgo03.ps.gz

Measurement and Analysis of - Runtime Profiling Data (Correct)
To examine the effect of compiler choice on such **profiles**, since this should be known when gathering
test suites and compilers. Keywords Java **Virtual Machine**, Bytecode Analysis, Contingency Measure 1.
optimisations such as Just-In-Time (JIT) 8] and **hotspot**-based [2] compilation, as well as comparative
www.cs.ucsb.edu/~ckrintz/papers/javaprof-scam01.pdf.gz

WS 9. The First International Workshop on - Unanticipated Software Evolution (2002) (Correct)
bytecode instrumentation allows developers to **profile** dynamically selected parts of a target
supported by Sun Microsystems' Java **HotSpot Virtual Machine**: Pazandak [21] presents ProbeMeister, a
in JDK 1.4 and supported by Sun Microsystems' Java **HotSpot Virtual Machine**: Pazandak [21] presents
www.informatik.uni-bonn.de/~gk/papers/ecoop2002/ecoopWsReportUSE2002.pdf

Cache Performance in Java Virtual Machines: A Study of.. - Rajan, Rubio, John (2002) (Correct)
the mixed mode execution engine [19]which uses **profile** based feedback to interpret/compile bytecodes.
1 Cache Performance in Java **Virtual Machines**: A Study of Constituent Phases Anand S.
has been shown to be comparable to Sun's JDK 1.3 (**HotSpot**) VM. Latte boasts of a highly optimized JIT
www.ece.utexas.edu/projects/ece/lca/ps/anand.pdf

Improving Branch Predictability in Java Processing - Li, John, Narayanan (2001) (Correct)
a complete system simulation environment, we **profile**, analyze and quantify performance issues of
format known as bytecodes and a Java **Virtual Machine** (JVM) Lind99] dedicated to a specific
in state-of-the-art JVM technologies such as Sun's **HotSpot** **[HotSpot99]**The execution of Java applications
www.ece.utexas.edu/projects/ece/lca/ps/tao-TR-feburary-2001.pdf

Platform Independent Dynamic Java Virtual Machine Analysis.. - Gregg, Power, al. (2001) (Correct)
a platform independent analysis of the dynamic **profiles** of Java programs when executing on the Java
Platform Independent Dynamic Java **Virtual Machine** Analysis: the Java Grande Forum Benchmark
such as Just-In-Time (JIT) e.g. 1, 8]and **hotspot**-centered compilation (see [9] for a survey)The

aspen.ucs.indiana.edu/CandCPandE/jg2001/C573waldron/c573jour1.pdf

Measurement and Analysis of Runtime Profiling Data for.. - Horgan, Power, Waldron (2001) (Correct)
does not provide full information on the dynamic **profile** of the program. For large programs the dynamic
by the execution of the code on the Java **Virtual Machine** (JVM)The problem addressed by this
yielding techniques such as Just-In-Time (JIT) and **hotspot**-centered compilation [2]However, the
www.cs.may.ie/~pgibson/TRseries/nuim-cs-tr-2001-04.pdf

Hardware Support for Profiling Java Programs - Hanish, Cohen (Correct)
The JIT system generates in-line machine code to **profile** each method. In the prologue of the method, the
system has been developed for the Kaffe Java **Virtual Machine**'s (JVM) Just-In-Time (JIT) i386 translator
code as proposed by Hlzle [3] and implemented in **HotSpot** [6]This profiling information is essential
www.eb.uah.edu/~cohen/javanauts/papers/iccd99b.ps

Efficient Parallel Execution of Irregular Recursive Programs - Prechelt, Hänßgen (1999) (Correct)
code even for irregular problems whose parallelism **profile** is highly data-dependent and can hence not be
on profiling are used by Sun's **Hotspot** Java **Virtual Machine** 1 With respect to the goals of REAPAR,
optimizations based on profiling are used by Sun's **Hotspot** Java **Virtual Machine** 1 With respect to the
wwwipd.ira.uka.de/~prechelt/Biblio/reapar_tpds1999.ps.gz

Try your query at: Google (CiteSeer) Google (Web) Yahoo! MSN CSB DBLP

Web | Images | Video | Directory | Local | News | Products

YAHOO! SEARCH profile "virtual machine" hotspot "cache miss" Search

My Web BETA Shortcuts Advanced Search Preferences

Search Results Results **1** - **8** of about **8** for profile "virtual machine" hotspot "cache miss" - 0.35 sec. (About this pa

1. http://www.cse.psu.edu/~pagrawal/Courses_Taken/521/Second_Report.doc (MICROSOFT WORD)
... initially started with **Hotspot** Java **Virtual Machine** (JVM) but later ... BIT classes to instrument and **profile** the original code ... stack and the **cache miss** ratio. Our modification builds ...
www.cse.psu.edu/~pagrawal/Courses_Taken/521/Second_Report.doc - 40k - View as html - More from this site

2. Dynamic Memory Layout Transformations for Java (PDF)
... started with **Hotspot** Java **Virtual Machine** (JVM) but later on ... the BIT classes to instrument and. **profile** the original code ... Whenever the **cache miss** ratio reaches a threshold value, this ...
www.cse.psu.edu/~pagrawal/Courses_Taken/521/Second_Report.pdf - 96k - View as html - More from this site

3. Modularity, Hardware Profiling, and Mixed ISA Execution in Managed Runtimes (PDF)
... defined interfaces and modularity. Core **Virtual Machine** (VM) ... HW based **Hotspot** detection. Integrated **profile** creation, mgmt & use ... Abstracts hardware **cache miss** sampling up to metadata ...
www.research.ibm.com/vee04/Srinivas.pdf - 110k - View as html - More from this site

4. Instruction-Set Simulation and Tracing
Instruction-Level Simulation And Tracing. This is $Revision: 1.107 $, last updated $Date: 2004/06/08 17:36:44 $. For an up-to-date version, please check www.xsim.com/bib.
www.xsim.com/bib/index1.d/Index.html - 318k - Cached - More from this site

5. -C H A P T E R
Performance Analyzer. C H A P T E R 5. Understanding the Performance Analyzer and Its Data. The Performance Analyzer reads the event data that is collected by the Collector and converts it into performance metrics. ... data files representing the **profile** events in the life ... by the Java **HotSpot virtual machine**, and are referred ... handling, memory management, and Java **HotSpot virtual machine** compilation ...
docs.sun.com/source/817-5068/AdvancedTopics.html - 143k - Cached - More from this site

6. 7 - C H A P T E R -
... a small number of **profile** packets is recorded, the call ... compiled by the Java **HotSpot virtual machine**, and there is ... such as at a **cache miss** or a resource queue ...
docs.sun.com/source/816-2458/AdvancedTopics.html - 106k - Cached - More from this site

7. Digital Mars - D - Java vs C Benchmarks
... Java virtual machines do some amazing things: they **profile** the ... The people working on **hotspot** in particular have hinted that Java ... Actually, the **virtual machine** has great overhead ...
www.digitalmars.com/d/archives/13897.html - 93k - Cached - More from this site

8. Java Technology and XML-Part 3: Performance Improvement Tips
... Network Community. **Profile** and Registration | Why Register? ... TRACE) { System.err.println("Query **cache miss** (mismatch)."); ... Asked Questions about the Java **HotSpot Virtual Machine** ...
jdl.sun.com/developer/technicalArticles/xml/JavaTechandXML_part3 - 77k - Cached - More from this site

---

Web | Images | Video | Directory | Local | News | Products
Your Search: profile "virtual machine" hotspot "cache miss" Search